IOSR Journal of Engineering
Apr. 2012, Vol. 2(4) pp: 726-730

An Evolutionary Neural Network Architecture Optimization Algorithm for Hand Written Digits Recognition

S. M.Krishna Ganesh[1], P. Saranya[2]
[1]*Department of CSE, SJUIT, Tanzania*
[2]*Department of CSE, Kalasalingam University, India*

ABSTRACT

Artificial Neural Network model is inspired by the functioning of the human brain. It is configured for a specific application, such as pattern recognition or data classification, through a learning process. In this paper we present a novel neural network architecture optimization algorithm from the given training pattern sets to solve a specific problem based on the principles of neuro-biology. Generally the neural network used for most of the applications are fully interconnected and bound to contain superfluous links and units. These superfluous links and units will contribute additional computational burden and also degrade its performance.. Our algorithm effectively removes all non-contributory links by directly using the characteristics of the given set of training pattern and determines the number of hidden layer neurons that can provide the better performance. We have demonstrated the working of our algorithm. The simulation results shows that our algorithm out performs other existing pruning algorithm for hand written digits recognition problems The various performance related graphs shows that our algorithm is computationally and performance wise out performs when compared than other existing pruning algorithms. The best feature of our algorithm is that it incorporated both pruning and growth strategies..

Keywords **– Architecture Optimization, Contributory links, Handwritten digit recognition and Neural network.**

I. INTRODUCTION

In most of the applications, neural network has been successfully applied to solve many different problems, including dynamic system identification, pattern classification, and adaptive control. As we know, before a neural network can be employed, its dimensions like number of layers, number of neurons in each layer and how they are connected must be predetermined. However, choosing the appropriate size of a neural network is a very difficult task and often comes down to guess work. In general, small-sized networks, even though they show good generalization performance, tend to fail to learn the training data within a given error bound, whereas large-sized networks learn easily the training data but yield poor generalization, unnecessary arithmetic calculations and high computation cost. For real time applications, the reduction of network size may save as precious hardware implementation time. In solving the problem of defining an optimal network topology, two main suggestions emerge. First, it is possible to decide on a relatively large network and then prune superfluous or redundant connections. This process is called as "neural network pruning". By using pruning techniques to determine the ideal size of a network can result in network topologies that never reach acceptable levels of accuracy on some classification problems**.**The second possibility is to start with a very small network and add nodes to grow it as necessary to learn a problem. This type of network grows as it learns by installing fully connected nodes into the network topology. This is called as "neural network growing". One drawback of this solution is the creation of a large interconnected system that is very deep in layers. In a fully interconnected neural network, the neurons of a particular layer are connected to the corresponding neurons and their neighbors in the other layers. Mostly, the present day neural network for solving a given problem does not provide an efficient performance due to superfluous interconnecting links. So there is a need arise to prune these unwanted superfluous links and consequently all the non-participating hidden layer neurons.Most of the existing pruning algorithms using the characteristics of the training pattern in indirect and complex way to prune the neural network are not definitely perfect because of that these algorithms prune some links (Contributory links) that are useful to improve the generalization ability and leaves some links (Non contributory links) that are not useful by randomly redistributing the link weights. It is evident from the above facts that these algorithms may degrade the generalization ability of the neural network and do not provide a perfectly fitting architecture for the given training pattern set and provide optimal performance. The above finding gave us a clue to develop a new systematic approach to prune only the non-contributory links and neurons, by directly using the characteristics of the given set of training pattern. Our novel algorithm will give rise to fully optimized neural network architecture for a given training pattern set with enhanced performance ability

II. MOTIVATION

The total number of neurons in the human brain is about 100 billion and each has connected to nearly 10,000 other cells. This means that each neuron may sends and receives impulse from as many as 10,000 target cells. The conclusions drawn from biological model are Biological neural networks are not fully inter-connected, The average number of interconnections per unit is only 10,000, When one cell repeatedly assists in firing another, the axon of the first cell develops synaptic knobs (or enlarge them if they already exists) in contact with the some of the second cell, Our brain takes raw sensory data from sensory organs and

transfers it into percepts, which correspond, to mental conceptual categories. Any data that does not conform to this requirement of perceptual organization cannot be processed. Even when data is organized into perception the judgment made by the brain is sometime internally contradictory. The above biological details motivated us to design this novel method to optimize neural network architecture using the given set of training patterns. This approach is called as evolutionary neural network architecture optimization algorithm.

III. RELATED WORK

A rule of thumb for obtaining good generalization in systems is that one should use the smallest system that will fit the data. A typical neural network contains an input layer, an output layer, and one or more hidden layers. The number of outputs and inputs are usually fixed; while the number of hidden layers and number of hidden neurons in each hidden layer can be varied. In this paper, we focus on the studies of pruning algorithms for multi-layer feedforward neural networks. The simplest way to find the optimum network size is to use a brute force approach that produces all the combinations of networks within a desirable range, trains them, and then chooses the best one. This process is usually not an efficient way to solve the problem. There are two another ways to perform the neural network optimization which is growth algorithm and pruning algorithm. Pruning algorithm will be again classified into two categories: Sensitivity algorithm force to estimate the units or links are least important and deletes them during training i.e., the network is trained, sensitivities are estimated, and then weights or nodes are removed Penalty term algorithm will modify the cost function so that back propagation based on the function drives unnecessary weights to zero and, in effect, removes them during training. Even if the weights are not actually removed, the network acts like a smaller system. Chung F. L. and Lee T. offered Network-growth approach to design of feedforward neural networks. A network-growth approach is pursued to address the problems concurrently and a progressive-training (PT) algorithm is proposed. The algorithm starts training with a one-hidden-node network and a one-pattern training subset. The training subset is then expanded by including one more pattern and the previously trained network, with or without a new hidden node grown, is trained again to cater for the new pattern. Such a process continues until all the available training patterns have been taken into account. At each training stage, convergence is guaranteed and at most one hidden node is added to the previously trained network.By using magnitude based pruning algorithm, John Sum dealt with Comparative study on various pruning algorithms for RNN I: Complexity Analysis and told that several non-heuristic pruning algorithms for fully connected RNN will be investigated, some of them are extended from heuristic based approaches and some of them are based on weight magnitude, together with some tricks on the pruning procedures. Because of high computation cost and computational complexity by using heuristic approach, propose the idea of non-heuristic algorithms is the inclusion of skipping and re-pruning. By using weight magnitude, whole list of weight have been checked, the pruning process re-run again and again until no more weight can be removed. As pruning a recurrent neural network is already a difficult problem, skipping and re-pruning does not introduce much overhead. Optimal Brain Damage algorithm assumes that the Hessian matrix (H) is diagonal. That is equivalent to assume that the total change in E when several weights (W) are deleted is the sum of δE caused by deleting each of the weights individually. Basically, the saliency is approximated by the second derivative of the cost function with respect to the weight. By using this algorithm, Manabu Kotani, Akihiro Kajiki and Kenzo Akazawa have proposed A structural learning algorithm for multi-layered neural networks for organizing the structure of the multi-layered neural networks. The proposed pruning algorithms consists of two already known algorithms such as the structural learning algorithm with forgetting and the optimal brain damage algorithm using the second derivatives of the assessment for pruning. It can able to find the set of weights whose pruning will cause the least increase of the object function which includes only the error term. After the network is slimmed by the structural learning algorithm with forgetting, unimportant weights are pruned from the network using the second derivatives. Babak Hassibi and David G. Stork provide general network pruning to present the OBS algorithm. Unlike OBD, OBS does not only delete a single weight, say, wj, but it will also adjust the remaining weights optimally to give the least increase in the error function by the following formula It is significantly better since it prunes more weights and yields good generalization of data than magnitude-based methods and Optimal Brain Damage which often remove the wrong weights. There is no restrictive assumption about the form of the network‟s Hessian as in OBD algorithm. It does not demand retraining after the pruning of a weight. Crucial to OBS is a recursion relation for calculating the inverse Hessian matrix from training data and structural information of the net.

IV. NOVEL ANN ARCHITECTURE

In fully connected neural networks only a group of links will contribute to learning process we name them as contributory links and the rest of the links as non-contributory links. We again classify the non-contributory links in two categories. The first category of links does not take part in the learning process; the second category of links takes part in the learning process and does not contribute anything significantly during the learning process. Moreover the convergence of the neural network during the learning process requires some minimum number of processing neurons in the hidden layer. So we divided the process of constructing the optimal architecture (i.e. the minimum number of interconnection links and the hidden layer neurons which are required to learn the given input pattern without any mistake) in three stages. The first stage is identifying the contributory links. We have used the given set of training pattern to identify these links. The second stage is determining the minimal number of hidden layer neurons. This iterative and time consuming procedure but can be stopped once the network is converged and before resulting in fully optimized neural network if it is desired so. The minimal number of hidden layer unit found in the above step is then adjusted to get optimal performance architecture

with respect to defined performance criteria. Our computation technique combines characteristics both the pruning and growth models and use the performance criteria. The steps involved to optimize the neural network architecture are given below.

Step 1: Trees are generated from units of first training set pattern in the output layer neurons as roots. Then the root of each tree is connected to all the hidden layer neurons as immediate children. Each child in turn connected to the activated input neurons.

Step 2: The trees generated in step1 are superimposed to create the links between the neurons.

Step 3: The above two steps repeated for all training patterns in the training set.

Step 4: Remove all the neurons that are not connected with any links.

Step 5: Generate various performance-related graphs to determine the optimal hidden layer units.

Finally the network is generated using above set gives rise to minimal number of interconnection links for the given training pattern and gives optimal performance. Matrices are used to store the links between neurons generated by this method.

To demonstrate pruning of superfluous links we will take a simple example of table 1.

	Training pattern 1	Training pattern 2
Input	100	101
Target	111	101

The above table provides the size of both the input and output pattern. (If size of input layer unit and Output layer unit are selected arbitrarily then algorithm will determine hidden layer neurons) These details are used to determine the number of input layer neurons that is 3 and output layer neurons is 3 and to calculate the initial number of hidden layer neurons that is 2. The network building stages are shown from figure-1 to Figure-7.

Starting network without any link

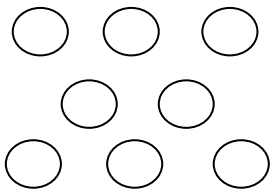


Figure : Network Building Stages

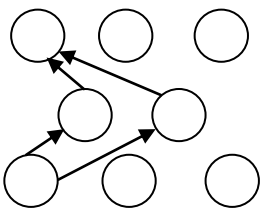


Figure : Network Building Stages

Tree Generated from a second activated unit off pattern 1

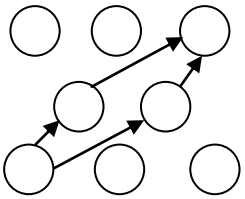


Figure : Network Building Stages

Super imposition of trees generated from pattern 1

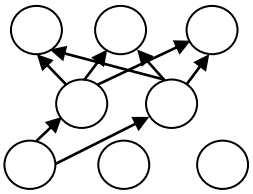


Figure : Network Building Stages

Super imposition of trees generated from pattern 2

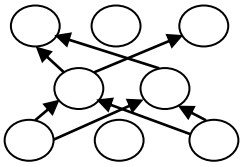
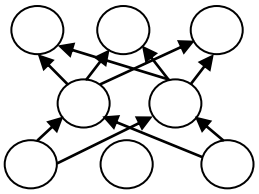


Figure : Network Building Stages

Optimized neural network architecture



V. PROPOSED NEW PRUNING ALGORITHM

Given:
The size of input pattern of „M" bits
The size of output pattern is „N" bits
The total number of training patterns is „P"

Evolutionary Neural network architecture Optimization Algorithm

H←(M+N)/2
Done_SI←False
Done_SD←False
For I←1 to P
do For J←1 to N
do If(O_PAT_VAL[I,J]=1)
then For K←1 to H
do If(there is not link (output_unit[J], hidden_unit[K]))
then create link link (output_unit[J], hidden_unit[K])
For L←1 to M
Do If (I_PAT_VAL[I,L]=1)
Then If(there is not link (input_unit[L], hidden_unit[K]))

Then create link (input_unit[L], hidden_unit[K])
if(Neural Network is converged)
then Done_SI←True
If Done_SD=False
Then H←H-1 and Goto Step 4
Else H←H+1 and Stop
Else Done_SD←True
Then If(Done_SI=False)
Then H←H+1 and Goto step 4
Else
H←H-1 and stop

VI. IMPLEMENTATION AND RESULTS

We have chosen handwritten digits recognition problem for our experiments. The special significance of selecting this problem is its usefulness in automated postal mail sorting by reliably recognizing the Zip (pin) Code. We have downloaded the data from website http://www.cs.toronto.edu/~roweis/data.html.These data sets are in image format and needs preprocessing in order to convert into acceptable neural network input format.. In the preprocessing, first, these digits images are converted into standard size (16X16) and then converted into binary images. Finally the pixel values of the images are used as input to the network. The output is a vector of 10 bits where each bits represent one of the ten digits. After preprocessing we have carefully selected for creating training data set and testing data set. Our training data set was created such a way that it include every variation in the hand written digit image for all ten different digits. The test data set again carefully selected in such way that it include slight variation from training data image. This is very useful in testing the generalization ability of the neural network. We have use Stuggart Neural Network Simulator(SNNS) for conducting our experiments. This is very flexible simulator for conducting experiments for many research problems. A small C program was written to identify the non-contributing links. The neural network was constructed by bignet option of SNNS simulator with 256 input units, 10 output layer units and 133 initial hidden layer units. The interconnection links are made only for contributing connection. The optimal number of hidden layer units are determined by using our algorithm. A graph was drawn for number of hidden layer units versus network convergence time. From the Figure1 which shows the graph drawn between number of hidden layer units and converges time we can conclude that the network converges fast at 80 hidden layer units. So eights hidden layer units is the ***optima***l number hidden layer unit for this specific problem and selected data set. So we have kept eighty units in the hidden layer.

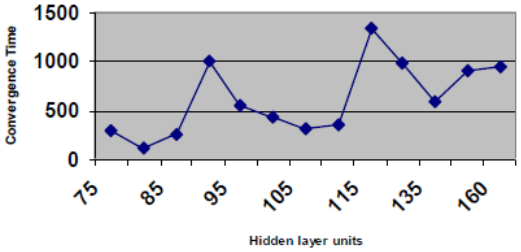


Figure: Convergence Time versus Number of hidden layer units

In order to check the convergence performance of our approach with respect to other approaches we have conducted experiment with similar our own bench mark training data set. A bar chart graph was drawn from the convergence taken by different approaches. It is seen very clearly from the chart (Figure2) that our approach has taken least convergence time.

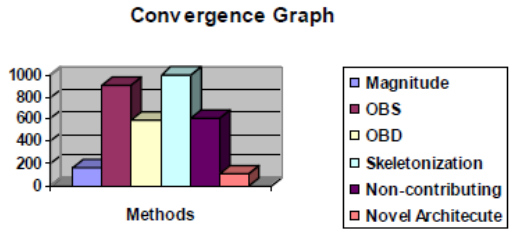


Figure: Methods Versus Convergence time (Training Cycles)

The final experiment was conducted to validate the performance of our algorithm compared to other algorithm with respect generalization ability of the neural network We have found that our algorithm out performs when compared to other approaches for our testing data set. The figure 3 shows the result of our experiment for determining the accuracy of predicting input data from the carefully selected testing data set.

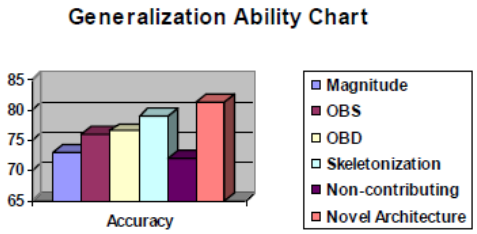


Figure: Comparison of generalization ability

VII. CONCLUSION

The superior performance of our proposed approach is due to careful removal non-contributing links and employing sufficient number of hidden layer unit. That is our architecture is neither over fit the input data nor under fit input data. Our approach inspired by neuro-biology principles provide almost right fitting of input training data set. This approach can extended for multiple hidden layer neural networks with more in depth study of various existing architecture, neuro-anatomy and neuro-physiology.

ACKNOWLEDGEMENTS

First of all we thank the almighty for giving us the knowledge and courage to complete the research work successfully. We express our gratitude to our respected Kalvivallal,T.Kalasalingam Founder, and Chancellor,Kalasalingam University, Krishnankoil, Srivilliputur,India and Dr. S.Radhakrishnan, Vice Chancellor,Kalasalingam University for allowing us to do the research work internally. We thank our friends and collegues for their support and encouragement.

REFERENCES

[1] S.Karthikeyan and Praveen Kumar Singh, "Performance based optimization of neural network architecture using training pattern sets," International Conference on Cognitive Science 2004, pp. 55–60.

[2] Devin Sabo and Xiao-Hua, "A new pruning algorithm for neural network dimension analysis," IEEE 2008, pp. 3313–3318.

[3] Nader Fnaiech, Sabeur Abid, Farhat Fnaiech and Mohamed Cheriet, "A modified version of a formal pruning algorithm based on local relative variance analysis," IEEE 2004, pp. 849-852.

[4] Manabu Kotani, Akihiro Kajiki and Kenzo Akazawa, "A structural learning algorithm for multi-layered neural networks," IEEE 1997, pp. 1105-1110.

[5] Eric Fock, Philippe Lauret and Thierry Mara, "A new saliency measure for inputs selection and node pruning in neural network," IEEE 2005, pp. 960-965.

[6] John Sum, "Comparative study on various pruning algorithms for RNN I: complexity analysis," in Proc . First international conference on machine learning and cybernetics, 2002, pp.2225-2230.

[7] Engelbrecht, A.P., "A new pruning heuristic based on variance analysis of sensitivity information", IEEE Transactions on Neural Networks 2001, Volume 12, Issue6, pp.1389-1399.

[8] Mozer, M.C. and Smolensky, P., "Skeletonization: A technique for trimming the fat from a network via relevance assessment," in Advance in Neural Information Processing 1989, D.S. Touretzky, Ed., pp.107-115.

[9] Ponnapalli, P.V.S., Ho, K.C., and Thomson, M., "A formal selection and pruning algorithm for feedforward artificial neural network optimization", IEEE Transactions on Neural Networks 1999, Volume10,Isssue 4, pp.964-968.

[10] Karnin, E.D., "A simple procedure for pruning back-propagation trained neural networks", IEEE Transactions on Neural Networks 1990, Volume 1,Issue 2, pp.239-242.

[12] Russell Reed, "Pruning Algorithms-A Survey", IEEE Transactions on Neural Networks, Volume 4, No.5, Sep 1993, pp. 740-747.

[13] Babak Hassibi, David G. Stork and Gregory J . IVolff, "0ptirna.l Brain Surgeon and General Xetlwork Pruning" ,IEEE 2008, pp.293-299.

[14] J.P. Thivierge, F. Rivest and T. R. Shultz, "A Dual-Phase Technique for Pruning Constructive Networks," IEEE 2003, pp. 559-564.

[15] Hubert Harrer and Josef A. Nossek, "Skeletonization: A New Application for Discrete-Time Cellular Neural Networks Using Time-Variant Templates," IEEE 1992, pp.2897-2900.

[16] Marsland, S., Nehmzow, S.U., and Shapiro, J., "A self-organizing network that grows when required", in Neural Networks, Volume 15, 2002, Issue 8-9, pp.1041-1058.

[17] Efe, M.O., Iplikci, S., Kayank, O., and Wilamowski, B., "An Algorithm for Fast Convergence in Training Neural Networks", International Joint Conference on Neural Networks, pp.1778-1782, Washington DC, July 15-19, 2001.